Metrics for Performance Evaluation of Distributed Application Execution in Ubiquitous Computing Environments

Prithwish Basu, Wang Ke, and Thomas D.C. Little
ECE Department, Boston University
{*pbasu,ke,tdcl*}*@bu.edu*

I. Introduction

With the rapid growth in tetherless portable computing and wireless networking technology, "Ubiquitous Computing" is coming of age. Corporate offices, college campuses, as well as hi-tech homes are embracing such technologies gradually. Specifically, wireless LAN support, which is one of the first concrete steps towards achieving ubiquity in computing has been growing at a fast pace, recently. A case in point is the Carnegie Mellon University campus where there are 400 wireless network access points enabling coverage for the entire campus, indoors and outdoors. As computing becomes truly ubiquitous, more resources are available to perform common user tasks (as well as more esoteric ones). This implies that in such environments, there are more choices of resources for performing these tasks. In this position paper, we describe specific metrics as well as system parameters that are necessary for quantifying success of dynamic execution of entire user applications or tasks in ubiquitous computing environments where different services are offered by several categories of devices.

II. What to Quantify?

We believe that efforts should be spent on quantifying the performance of the execution of a complete user task at hand, not just that of a certain service discovery step in the task. In ubiquitous computing environments, several devices, specialized or multipurpose, can participate in the execution of the task. Also, there are likely to be multiple occurrences of devices that offer similar services, hence a user need not be bothered about the choice of particular devices that participate in the task, as long as it is completed successfully. Therefore, the *quality* of the overall choice of devices that participate in a task needs to be quantified.

A user specifies the structure of an application by means of a *resource graph* consisting of *nodes* representing "logical" computing resources and *edges* representing data flow dependencies or other requirements such as physical proximity between nodes, to facilitate the application [1]. At runtime, particular instances of devices/resources need to be selected from among multiple available instances for efficient execution of the given application. We refer to this process as *Application Mapping* or *Instantiation*. Note that the issues here are different from those in simple service discovery since we are concerned with the discovery of all resources needed for the execution of the entire application while obeying the constraints specified in the resource graph. We also need to measure the performance of the actual execution process as a function of several system parameters.

III. Metrics for Performance Evaluation

In an application execution framework mentioned is Section II, it is essential to lay down system level metrics which can quantify success or failure of achieving a particular ubiquitous computing task. We list some metrics, which we believe can help in the process of performance evaluation of such applications, below.

1. **Application mapping time**: Time taken to map an application characterized by a resource graph onto a set of suitable devices which will actually execute the application.
2. **Application mapping efficiency**: The following submetrics can evaluate different aspects of the mapping efficiency.
 (a) **Average/Maximum Dilation**: Each edge of the resource graph can get mapped onto a multi-hop shortest path on the ubicomp network. This measures the average/maximum stretch when the mapping is completed. Obviously, this value is greater than 1. Now, each edge of the resource graph can have weights signifying requirements for that particular association. These weights are functions of characteristics such as data rate, relative importance in overall task, error tolerance etc. A low dilation mapping is good since if most required devices are located nearby, the chances of achieving high overall throughput are greater.
 (b) **Node Congestion**: A mapping with low dilation can suffer from bottleneck paths passing through a single device or a few devices. Minimization of this metric encourages mappings with lesser number of paths passing through bottleneck devices. Hence this automatically helps in load balancing.

(c) **Messaging overhead**: Number of messages exchanged between devices during the establishment of a mapping. Broadcast based solutions tend to have a high message overhead although they are usually better suited for highly mobile networks.

3. **Resilience of Application mapping to Device Mobility**: Mobility of devices can cause network partitions and therefore, application outages. The following set of metrics measure the impact of device mobility on an application mapping.

(a) **Frequency of Application Disruption**: The rate at which an application is disrupted due to network partitions.

(b) **Application Recovery Time**: Time taken to discover replacement devices for re-mapping the affected parts of the resource graph + time to perform *state migration* to these new devices.

(c) **Average Connected time**: Time for which an application runs without any disruptions.

4. **Application performance after mapping**: These set of metrics attempt to measure how well an application executes after all the required resources have been discovered and instantiated.

(a) **Average Effective Throughput**: Fraction of ADUs received by intended recipients of the ADUs that were supposed to be received under perfect network conditions. This is a normalized metric and can attain a maximum value of 1. Mobility of devices can cause network partitions which can result in task disruptions, hence lowering this. This can be an indicator of data loss.

(b) **Average number of Re-transmissions**: Since application data can be lost because of disruptions, buffering and re-transmissions may be required at data sources. This metric measures the number of times a data unit is needed to be re-transmitted before successful reception by an intended instantiated recipient.

(c) **Average Data Delay**: The amount of time that elapsed between the sending of a data unit and its successful reception at an intended recipient. This may include the recovery time after a disruption.

IV. Variation of System Parameters

For every application, a subset of the above metrics should be measured for a combination of values of the following system parameters (whichever are relevant):

1. Number of devices in the network – this relates to area density, richness of network connectivity etc.
2. Number of devices that need to participate in a particular application (number of nodes in the resource graph).
3. Complexity of relationships between nodes of the resource graph of an application (number of edges in the resource graph, structure of the resource graph - whether it is a tree or not)
4. Number of instances of resources in the ubicomp network with similar capabilities or attributes.
5. Number of different task instances (of same or different tasks) running simultaneously on the ubicomp network.
6. Fraction of the devices in the network that rely on a wireless access point for routing (rest use ad hoc routing).
7. Average data rate required by the distributed application.
8. Background traffic patterns ranging from low load to heavy load.
9. Mobility patterns of users and other service providing devices ranging from random to highly predictable. Particular parameters in this context are: (a) Speed of motion, and (b) Frequency and duration of pauses in mobility.

V. Conclusion

In this position paper, we presented several application level metrics for evaluating the performance of a distributed task on a ubicomp network. We also described some relevant system parameters that should be varied during the experimentation process. We note that different distributed applications on a ubicomp network may have different levels of tolerance with respect to the metrics proposed in this paper. e.g., some applications may be able to tolerate large delays due to re-mappings but not data loss. Hence a designer of a particular application should keep in mind the relative importance of these metrics for that application.

References

[1] P. Basu, W. Ke, and T.D.C. Little, “A Novel Task Based Approach for Supporting Distributed Applications on Mobile Ad Hoc Networks,” *BU MCL Technical Report, TR-07222001,* July 2001. *http://dazzler.bu.edu/∼pbasu/research/TR-07222001.ps.gz*